HOW TO COMPUTE THE CONSTANT TERM OF A POWER OF A LAURENT POLYNOMIAL EFFICIENTLY

PAVEL METELITSYN

ABSTRACT. We present an algorithm for efficient computation of the constant term of a power of a multivariate Laurent polynomial. The algorithm is based on univariate interpolation, does not require the storage of intermediate data and can be easily parallelized. As an application we compute the power series expansion of the principal period of some toric Calabi-Yau varieties and find previously unknown differential operators of Calabi-Yau type.

1. INTRODUCTION

The constant terms of powers of Laurent polynomials appear in different areas of mathematics and theoretical physics [D], [DvK]. They show up prominently in the expansion of the *fundamental period*

$$\Phi(z) = \frac{1}{(2\pi i)^N} \int_\Gamma \frac{1}{1 - zf(X)} \frac{dX_1}{X_1} \frac{dX_2}{X_2} \cdots \frac{dX_N}{X_N},$$

of a Laurent polynomial in $X_1, \ldots, X_N$, and Γ is the N-dimensional torus cycle $|X_i| = \epsilon$. In the theory of mirror symmetry for Calabi-Yau hypersurfaces in a toric variety, this fundamental period is of great importance, [BK], [BvS]. To compute the Taylor expansion of Φ in the neighborhood of $z = 0$ we write

$$\frac{1}{1 - zf(X)} = \sum_{i \geq 0} (f(X))^i z^i$$

and integrate using Cauchy's formula. The n-th coefficient in the power series expansion of Φ is therefore the constant term of the n-th power of the Laurent polynomial f. For example the constant term of the 151st power of

$$\begin{aligned} f &= \frac{X}{YZ} + \frac{ZT}{XY} + ZT + T + \frac{Y}{X} + \frac{Z}{X} + \frac{Y}{XT} + Y + \frac{XY}{ZT} + \frac{X}{Z} + \frac{Y}{T} \\ &+ Z + \frac{1}{T} + X + \frac{X}{Y} + \frac{ZT}{Y} + \frac{X}{ZT} + \frac{Y}{ZT} + \frac{1}{X} + \frac{1}{Y} + \frac{ZT}{X} + \frac{T}{Y} + \frac{1}{Z} \end{aligned}$$

is
15412036066982883611159466717890839926274227993361685769096965357956125
08360971138505497488955831195692422950790226144730327544742024697381175810
3097074502829198076370950235391810731785760778732696320, a number with 200 digits.

In this paper we consider the following general problem:
How to compute for a multivariate (Laurent) polynomial f the coefficient of the monomial X^α, $\alpha = (\alpha_1, \ldots, \alpha_n)$ of the p-th power, $p \in \mathbb{N}$ of f.

At first glance this problem is trivial. Why can't we just split the problem in smaller ones and compute $f^r =: g = \sum b_\beta X^\beta$ and $f^s =: h = \sum c_\beta X^\beta$ s.t. $r+s = p$ and get the coefficient we are interested in by Cauchys product formula

$$\text{coeff. of } X^\alpha \text{ in } f^p = \sum_{i+j=\alpha} b_i c_j. \tag{1}$$

A general polynomial in n variables in which the degree of each individual variable is not greater than d has $(d+1)^n$ monomials. In our example $n = 4$, $p = 151$, $d = 2$ after multiplication with the common denominator $XYZT$. In order to use (1) we could compute f^{75} and f^{76} which will require to store about $2(2 \cdot 76 + 1)^4 \approx 10^9$ coefficients of intermediate polynomials g and h. While this number is not really big, we must keep in mind that, in general, the absolute value of the coefficients grow exponentially as p increases. Therefore, the amount of memory needed to store one particular coefficient will, typically, linearly grow with p. Another possibility is to use a multivariate discrete Fourier transform: evaluate the polynomial f on sufficently many points of the n-dimensional grid of roots of unity of sufficient high degree, compute the p-th power of the values and apply the inverse discrete Fourier transform to get the single coefficient we are interested in. Here, again, the number of interpolation nodes and therefore of the intermediate values which have to be stored is of the same order as the number of monomials in f^p i.e. $(2 \cdot 151)^4$.

In this paper, we present an efficient and, we believe, simple algorithm which adresses this problem. Our approach is similar to the DFT method in that it based on evaluation and iterpolation. However, the amount of memory needed is much lower and the running speed can be improved by a trivial parallelization of the evaluation. We also describe the implementation of the parallel version of the algorithm for the CUDA-enabled graphics hardware. Finally, an application to the problem of finding the Picard-Fuchs differential equation of the family of toric Calabi-Yau varieties is given. This problem was the main motivation for this work.

Acknowledgement: I like to thank D. van Straten for suggesting the problem and E. Schömer for useful discussions. This work was realised with support from the DFG priority program *Algorithmic and Experimental Methods in Algebra, Geometry and Number Theory*, as part of the project *Monodromy algorithms in* SINGULAR.

2. ALGORITHM

2.1. **Notations and Preliminaries.** Let f be a multivariate polynomial in variables $X_1, \dots, X_k$ with rational coefficients. Using multiindex notation we write

$$f = \sum_{i \in \Delta} c_i X^i,$$

$i := (i_1, \dots, i_k)$ and $\Delta \subset \mathbb{Z}^k$ a finite index set. Let $\deg_r(f)$ denote the degree of f in the variable X_r. For a given k-dimensional multiindex j we denote by $[f]_j$ the coefficient of f in front of X^j i.e.

$$[f]_j := c_j.$$

Furthermore let j' denote the $(k-1)$-dimensional multindex obtained by truncating j after the $(k-1)$-th component and for an r-dimensional multiindex i let ij denote the concatenation of both i.e. for $j = (j_1, \dots, j_k)$ and $i = (i_1, \dots, i_r)$

$$\begin{aligned} j' : &= (j_1, \dots, j_{k-1}), \\ ij : &= (i_1, \dots, i_r, j_1, \dots, j_k). \end{aligned}$$

Given $N+1$ pairs $(u_\alpha, w_\alpha) \in \mathbb{Q}^2$, $\alpha = 0, \dots, N$ such that $u_\alpha \neq u_\beta$ for $\alpha \neq \beta$, there is a unique *univariate interpolation polynomial* of degree N such that

$$\mathcal{I}_N(u_\alpha) = w_\alpha, \ \forall \alpha \in \{0, \dots, N\}$$

$$\mathcal{I}_N(u_0, \dots, u_N; w_0, \dots, w_N)(Y) = \sum_{i=0}^{N} w_i l_i(Y),$$

where

$$l_i(Y) \quad = \quad \prod_{i \neq j} \frac{Y - u_i}{u_j - u_i}$$

is the Lagrange basis polynomial for u_i. $\mathcal{I}_N$ has rational coeffiients and $\mathcal{I}_N(u_\alpha) = w_\alpha$, $\forall \alpha = 0, \dots, N$.

Let f_u denote the polynomial f with u substituted for the last variable i.e.

$$f_u(X_1, \dots, X_{n-1}) = f(X_1, \dots, X_{n-1}, u).$$

We have the following observation about the coefficients of the interpolation polynomial and those of f_u.

Lemma 1. *Let* $f \in \mathbb{Q}[X_1, \dots, X_n]$, $N := \deg_n(f)$, $\{(u_\alpha, w_\alpha)\}_{\alpha=0,\dots,N}$ *as above, and* $i = (i_1, \dots, i_n) \in \mathbb{N}^n$. *Then, we have*

$$[f]_i = \left[\mathcal{I}^N(u_0, \dots, u_N; [f_{u_0}]_{i'}, \dots, [f_{u_N}]_{i'})\right]_{i_n}. \tag{2}$$

Proof. Let

$$f(X_1, \dots, X_n) = \sum_{\nu_1, \nu_2, \dots, \nu_n} c_{\nu_1 \nu_2 \dots \nu_n} X_1^{\nu_1} X_2^{\nu_2} \dots X_n^{\nu_n}, \tag{3}$$

and

$$\mathcal{I}^N(u_0, \dots, u_N; [f_{u_0}]_{i'}, \dots, [f_{u_N}]_{i'})(Y) = \sum_{k=0}^{N} a_k Y^k. \tag{4}$$

Then, the right hand side of (2) equals a_{i_n} and, therefore, we have to show that $[f]_i = a_{i_n}$. The coefficients $a_0, \dots, a_n$ of the interpolation polynomial are found by multiplying the vector $([f_{u_0}]_{i'}, [f_{u_1}]_{i'}, \dots, [f_{u_N}]_{i'})^T$ by the inverse of the Vandermonde matrix for $u_0, \dots, u_N$. This inverse exists since $u_0, \dots, u_N$ are pairwise distinct. Therefore, we have

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} 1 & u_0 & u_0^2 & \cdots & u_0^N \\ 1 & u_1 & u_1^2 & \cdots & u_1^N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_N & u_N^2 & \cdots & u_N^N \end{pmatrix}^{-1} \begin{pmatrix} [f_{u_0}]_{i'} \\ [f_{u_1}]_{i'} \\ \vdots \\ [f_{u_N}]_{i'} \end{pmatrix} \tag{5}$$

By (3) we have

$$\begin{aligned}
[f_{u_j}]_{i'} &= \left[\sum_{\nu_1,\nu_2,\dots,\nu_{n-1},\tau} c_{\nu_1\nu_2\dots\tau} u_j^\tau X_1^{\nu_1} X_2^{\nu_2}\dots X_{n-1}^{\nu_{n-1}}\right]_{i'} \\
&= \sum_\tau c_{i_1 i_2 \dots i_{n-1}\tau} u_j^\tau \\
&= \sum_\tau c_{i'\tau} u_i^\tau,
\end{aligned}$$

where in the last expression $i'\tau$ is an abbriviration for the multiindex $(i_1,\dots,i_{n-1},\tau)$. Using this, rewrite (5) as

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} 1 & u_0 & u_0^2 & \cdots & u_0^N \\ 1 & u_1 & u_1^2 & \cdots & u_1^N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_N & u_N^2 & \cdots & u_N^N \end{pmatrix}^{-1} \begin{pmatrix} \sum_\tau c_{i'\tau} u_0^\tau \\ \sum_\tau c_{i'\tau} u_1^\tau \\ \vdots \\ \sum_\tau c_{i'\tau} u_N^\tau \end{pmatrix} \tag{6}$$

At the same time

$$\begin{pmatrix} \sum_\tau c_{i'\tau} u_0^\tau \\ \sum_\tau c_{i'\tau} u_1^\tau \\ \vdots \\ \sum_\tau c_{i'\tau} u_N^\tau \end{pmatrix} = \begin{pmatrix} 1 & u_0 & u_0^2 & \cdots & u_0^N \\ 1 & u_1 & u_1^2 & \cdots & u_1^N \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_N & u_N^2 & \cdots & u_N^N \end{pmatrix} \begin{pmatrix} c_{i'0} \\ c_{i'1} \\ \vdots \\ c_{i'N} \end{pmatrix}. \tag{7}$$

From (7) and (6) we get

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} c_{i'0} \\ c_{i'1} \\ \vdots \\ c_{i'N} \end{pmatrix}$$

and with (3)

$$a_{i_n} = c_{i' i_n} = c_i = [f]_i$$

follows. □

Since $\deg_n(f^p) = p \cdot \deg_n(f)$ we have

$$[f^p]_i = \left[\mathcal{I}^N(u_0,\dots,u_N; [f^p_{u_0}]_{i'},\dots,[f^p_{u_N}]_{i'})\right]_{i_n}, \tag{8}$$

where $N = p \cdot \deg_n(f)$.

2.2. **The algorithm and its application to the constant term series of a Laurent polynomial.** Let $h \in \mathbb{Q}[X_1^\pm,\dots,X_n^\pm]$ be a Laurent polynomial. We want to calculate the first coefficients of its constant terms series $([h^p]_0)_{p\in\mathbb{N}}$. Let $S = X_1^{s_1}\dots X_n^{s_k},\ s_i \geq 0$, be the common denominator of the Laurent monomials of h. Then $Sh \in \mathbb{Q}[X_1,\dots,X_k]$ and

$$[h^p]_0 = [S^p f^p]_{p\cdot s},$$

where $p \cdot s := (ps_1,\dots,ps_n)$. We want to apply (8) to $f = Sh$ and $i = p \cdot s$.

The simple idea now is to compute $\left[f^p_{u_j}\right]_{i'}$, $j = 0,\dots,N$ which appear in (8) recursively i.e.

$$\left[f^p_{u_j}\right]_{i'} = \left[\mathcal{I}^N(u_0,\dots,u_N; [f^p_{u_j u_0}]_{i''},\dots,[f^p_{u_j u_N}]_{i''})\right]_{i_{n-1}} \tag{9}$$

and so on for $f_{u_j u_i}, f_{u_j u_i u_k}, \dots$. With each level of recursion the number of variables decreases by one, therefore the depth of the recursion equals n. For simplicity assume

$$\deg_1(f) = \cdots = \deg_n(f) = d.$$

The computation of $[f^p]_i$ proceeds as follows:

(1) For $N = dp$ fix $u_0, \dots, u_N$ as in Lemma and compute $V^{-1} = (V_{ij})$, where V is the Vandermonde matrix of $(u_0, \dots, u_N)$.
(2) Invoke the recursive procedure $\mathrm{COEFF}(i, p, k, A)$ with initial parameters $k = n$ and $A =$"coefficient matrix of f".

The pseudo code of the procedure COEFF is given in the Algorithm 1.

Algorithm 1 $\mathrm{COEFF}(k, i, A, p)$

```
Input: $k > 0$, $i \in \mathbb{N}^k$, $A \in Mat(\underbrace{d \times \cdots \times d}_{k})$, $p \ge 1$
Output: $[f^p]_i$
1:  for all $0 \le s \le N$ do
2:      $B_{i_1,\dots,i_{k-1}} \leftarrow \sum_j A_{i_1,\dots,i_{k-1},j} u_s^j$ // Do it using Horner's rule
3:      if $k = 1$ then
4:          $w_s \leftarrow B^p$ // $B$ has $k - 1 = 0$ indices i.e. $B$ is a scalar
5:      else
6:          $w_s \leftarrow \mathrm{COEFF}(k - 1, i', B, p)$
7:      end if
8:  end for
9:  $S \leftarrow \sum_{s=0}^{N} w_s V_{i_k s}$ // Compute the $i_k$-th component of $V^{-1} w$
10: return $S$
```

To improve the performane we apply some well known computational tricks. The evaluation in the line 2 is done using the Horner's rule which for a polynomial $h(x) = a_0 + a_1 x + \cdots + a_l x^l$ at $x = x_0$ says

$$f(x_0) = a_0 + x_0(a_1 + x_0(a_2 + x_0(\dots))).$$

In this way the expensive computation of powers of x_0 is avoided. The Vandermonde matrix V can be inverted using classical Gauss elimination algorithm in $O(N^3)$ time. Since on each level of recursion we need only one row of V^{-1} (line 9 in the code above) precomputing the whole inverse seems too expensive. Fortunately, if the nodes $u_0, \dots, u_N$ are choosen to be equidistant there is a recursive algorithm which computes one single row of V^{-1} in $O(N^2)$ [T]. The computation of the p-th power in the line 4 can be done using binary powering algorithm as described in [K]. We also note that w_s's in line 6 do not depend on each other and thus the order in which they are computed does not matter. In particular they can be computed parallelly. Finally, the number of nodes N need not to be the same on each recursion level but depend on the degree of f in a particular variable i.e. we have to choose $N_k := \max\{\deg_1(f), \dots, \deg_k(f)\} \cdot p$ on the $(n-k)$-th recursion level. For example if f has degree 2 in one variable and degree 1 in the remaining variables we could improve speed by considering $2p$ interpolation nodes only on one level of recursion and p interpolation nodes on the ramaining.

If f happens to have special form

$$f = A + BX_1 + CX_1^2 \tag{10}$$

with $A, B, C \in \mathbb{Z}[X_2, \ldots, X_k]$ a trick can be applied to slightly speed up the computation. If (10) holds, then so is for f_u,

$$f_u = A_u + B_u X_1 + C_u X_1^2$$

with $A_u, B_u, C_u \in \mathbb{Z}[X_2, \ldots, X_{k-1}]$. Consequently, this decomposition holds on all levels of recursion. We note that in this case

$$f^p = \sum_{i,j} \binom{p}{i,j} A^i C^j B^{p-i-j} X_1^{p-i+j},$$

and

$$\begin{aligned} [f^p]_p &= \left[\sum_{i=0}^{p} \binom{p}{i,i} A^i C^i B^{p-2i} \right]_p \\ &= \sum_{i=0}^{p} \binom{p}{i,i} \left[A^i C^i B^{p-2i} \right]_p . \end{aligned} \tag{11}$$

since we need only the terms with $X_1^{p-i+j} = X_1^p$ i.e. $i = j$. We apply this ”shortcut” on the last but one level of recursion to compute $\left[f^p_{u_1,\ldots,u_{k-2}} \right]_p$ when we have to deal with polynomials in two variables. We use the modified version of COEFF and a new procedure SPLIT2 for this. Thus the depth of recursion is reduced by one. For the case $n = 4$ the call tree of the algorithm is depicted in Figure 2.2.

Algorithm 3 Modified version of COEFF

Input: $k > 0$, $i \in \mathbb{N}^k$, $A \in Mat(\underbrace{d \times \cdots \times d}_{k})$, $p \geq 1$

Output: $[f^p]_i$

```
 1: for all $0 \leq s \leq N$ do
 2:     $B_{i_1,\ldots,i_{k-1}} \leftarrow \sum_j A_{i_1,\ldots,i_{k-1},j} u_s^j$
 3:     if $k = 2$ then
 4:         $w_s \leftarrow \mathrm{SPLIT2}(i', B, p)$
 5:     else
 6:         $w_s \leftarrow \mathrm{COEFF}(k-1, i', B, p)$
 7:     end if
 8: end for
 9: $S \leftarrow \sum_{s=0}^{N} w_s V_{is}$
10: return $S$
```

The bottom procedure SPLIT2 is called exactely N^{n-2} times. Inside the SPLIT2 procedure we have two nested loops of depth 2. The bodies of both inner loops (lines 12-14 and 25-28 in the listing 4) are executed both at most $\frac{N^2}{2}$ times. Therefore the time complexity of the algorithm is of order $O(N^n)$. Thanks to SPLIT2 we gain a factor $\frac{1}{2^{n-2}}$ compared to the version which does not use SPLIT2. If we take into consideration the linear growth of the length of the numbers involved into

Algorithm 4 $\mathrm{SPLIT2}(i, A, p)$

```
Input: $k \geq 0, i \in \mathbb{N}^k, A \in Mat(\underbrace{d \times \cdots \times d}_{k}), p \geq 1$
Output: $[f^p]_i$
1:  $s \leftarrow 0$
2:  $m \leftarrow \frac{N}{2} + 1$
3:  for all $0 \leq i \leq N$ do
4:      $w_i \leftarrow (A_{00} + u_i(A_{01} + u_i A_{02})) \cdot (A_{20} + u_i(A_{21} + u_i A_{22}))$
5:      $t \leftarrow A_{10} + u_i(A_{11} + u_i A_{12})$
6:      if $n \equiv 0 \mod 2$ then
7:          $v_{m-1,i} \leftarrow 1$
8:      else
9:          $v_{m-1,i} \leftarrow t$
10:     end if
11:     $t \leftarrow t^2$
12:     for all $0 \leq j \leq m-2$ do
13:         $v_{m-2-j,i} \leftarrow t \cdot v_{m-1-j,i}$
14:     end for
15:     $w'_i \leftarrow 1$
16:     $z_i \leftarrow v_{0,i}$
17: end for
18: $t \leftarrow 0$
19: for all $0 \leq i \leq N$ do
20:     $s \leftarrow s + c_i z_i$
21: end for
22: $s \leftarrow M_0 \cdot s$
23: for all $1 \leq j \leq m$ do
24:     $t \leftarrow 0$
25:     for all $0 \leq i \leq N$ do
26:         $w'_i \leftarrow w'_i \cdot w_i$
27:         $t \leftarrow t + w'_i \cdot w_i \cdot c_i \cdot v_{j,i}$
28:     end for
29:     $s \leftarrow s + M_j \cdot t$
30: end for
31: return $s$
```

computation we arrive at assymtotical running time pN^n or $d^n p^{n+1}$, since N depends linearly on p and d.

As we can see from the description of the algorithm, the memory consumption is $O(N) = O(dp)$, which is the space needed to store the $n-1$ rows of the inverse Vandermonde matrix. In particular, for fixed d it is independent from the number of monomials f has.

Remark 1. When d is fixed, the running time depends on p and the number of monomials in f.

2.3. **Implemetation details.** As was mentioned before, the algorithm can be parallelised in a very straight-forward way. As a target platform we choose a graphics

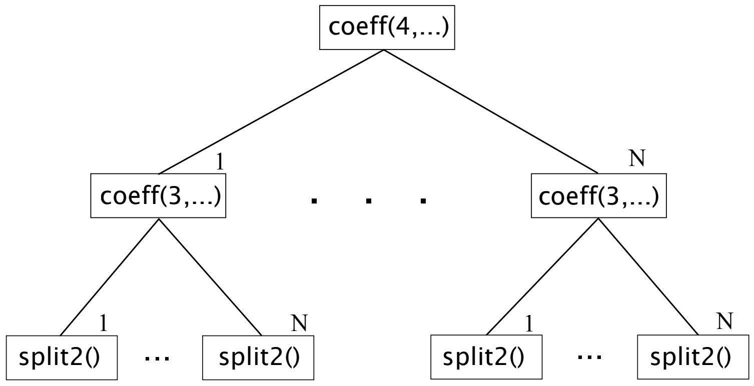

FIGURE 1. Call tree of the algorithm for $k = 4$.

card built around NVIDIA GeForce GTX 480 GPU[1]. This GPU is capable of running up to 480 parallel execution threads. The GPU is programmed using an extension of C/C++ programming language called *NVIDA CUDA*[2] *C* [N]. A typical CUDA program is executed on both the CPU and the GPU. The parts of the code that run on the graphics hardware are called *kernels* and each kernel can be run N times in parallel by N different CUDA threads. The threads are organized in one-, two- or three-dimensional *thread blocks* while blocks are organized in one- or two-dimensional *grid*. The graphics hardware was origianlly created for the purpose of low-precision floating point operations it uveils its full computing power only when dealing with 32-bit floating point numbers. Therefore we can perform exact computations only on numbers within the range $-2^{23} \dots 2^{23}$ or equivalently $-8388608 \dots 8388608$. That is because only 23 bits of 32 are used to store the significant digits (8-bits being reserved for the exponent and one remaining bit for the sign). This range is far too small to be useful due to the rapid growth of the numbers $[f^p]_0$. Therefore we used a *residue number system* (RNS) to represent large integers. Given a (fixed) set of pairwise coprime natural numbers $m_1, \dots, m_s$, the integer $x < M := \prod m_i$ is represented by the system of residues

$$\begin{aligned} x_1 : &= x \mod m_1 \\ &\vdots \\ x_s : &= x \mod m_s, \end{aligned}$$

M is called the *dynamic range* of the system. By the well-known *Chinese Reminder Theorem* the number x can be reconstructed from $(x_1, \dots, x_s; m_1, \dots, m_s)$. In practice, the conversion from an RNS representation of x to a decimal representation is done with the *Mixed Radix Conversion (MRC)* i.e. x is represented in the form

$$x = a_1 + a_2 m_1 + a_3 m_1 m_2 + \cdots + a_s m_1 m_2 \dots m_{s-1},$$

[1]Graphics Processing Unit
[2]Compute Unified Device Architecture

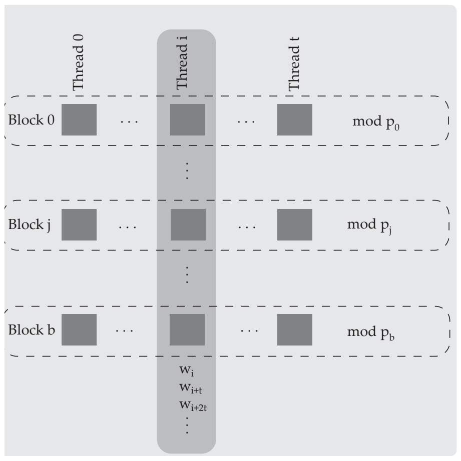

FIGURE 2. Arrangement of threads.

where a_i's are called *mixed radix digits*. Once a_i's are known the decimal value can be computed easily. One classical algorithm for finding a_i's from RNS representation of x is given in [SR].

In our implementation the program branches in several execution threads in the procedure COEFF on the top level of recursion. The threads are organized in b blocks with t threads per block. All the threads within the k-th block performs computation modulo prime number p_k. The i-th thread of k-th block computes $w_j \mod p_k$ for $j = i \mod t$. The arrangement of threads is illustrated in Figure 2.

3. APPLICATION TO TORIC CALABI-YAU VARIETIES

In the theory of toric Calabi-Yau varieties the constant terms of powers of Laurent polynomials occur in the following way. Consider a reflexive lattice polytope $\Delta \subset \mathbb{R}^4$. Every vertex $v_i = (v_{i,1}, \dots, v_{i,4}) \in \mathbb{Z}^4$ of Δ corresponds to a monomial $\prod_{j=1}^4 X_j^{v_{i,j}}$. Then the Laurent polynomial corresponding to Δ is

$$f_\Delta = \sum_{i=1}^{k} \prod_{j=1}^{4} X_j^{v_{i,j}}.$$

This polynomial defines a family of hypersurfaces $X_z := \{1 - z \cdot f_\Delta = 0\} \subset (\mathbb{C}^\times)^4$. The function

$$\Phi(z) = \sum_{i=0}^{\infty} \left[f_\Delta^i\right]_0 z^i \tag{12}$$

is the period of the holomorphic 3-form

$$\omega_z = \mathrm{Res}_{X_z}\left(\frac{1}{1 - z f_\Delta(X)} \frac{dX_1}{X_1} \wedge \frac{dX_2}{X_2} \wedge \frac{dX_3}{X_3} \wedge \frac{dX_4}{X_4}\right)$$

on X_z and is a solution of the Picard-Fuchs equation

$$L\Phi(z) \quad = \quad 0$$

where

$$L = P_0(\theta) + zP_1(\theta) + \cdots + z^k P_k(\theta), \tag{13}$$

$\theta := z\frac{d}{dz}$ and P_i are polynomials with integral coefficients

$$P_i = \sum_{j=0}^{d_i} b_{ij} X^j.$$

The numbers $(a_i)_{i\in\mathbb{Z}}$

$$a_i := \begin{cases} \left[f^i\right]_0, & i \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

then satisfy the following recurrence relation

$$P_0(n)a_n + P_1(n-1)a_{n-1} + \cdots + P_k(n-k)a_{n-k} = 0, \tag{15}$$

for all $n \in \mathbb{N}_0$. We say that the recurrence has length $k+1$ and degree $d := \max\{\deg P_i\}$. Conversely, if a sequence satisfies (15) then its generating function is anihilated by the differential operator (13).

Given Δ we want to find the corresponding differential operator L. To do this, we compute first N coefficients of the constant terms series (12) of f_Δ and try to find a reccurence of the form (15). Assume there is one with lenght $k+1$ and degree d. Then it is determined by $(k+1)(d+1)$ coefficients $b_{ij},\ i = 0,\dots,k,\ j = 0,\dots,d$ of $P_0,\dots,P_k$. We check if there is a solution to system of $N+1$ linear equations for the $(k+1)(d+1)$ unknowns b_{ij}

$$\begin{cases} P_0(N)a_N + P_1(N-1)a_{N-1} + \cdots + P_k(N-k)a_{N-k} & = & 0 \\ \vdots & \vdots & \vdots \\ P_0(2)a_2 + P_1(1)a_1 + P_2(0)a_0 & = & 0 \\ P_0(1)a_1 + P_1(0)a_0 & = & 0 \\ P_0(0)a_0 & = & 0 \end{cases} \tag{16}$$

We have to keep this system overdetermined i.e. to have $N+1 > (k+1)(d+1)$. If there is a solution which does not change when we add more equations, we hope that we found the correct operator. Since, a priori, we do not know the length and the degree of the reccurence we make the Ansatz (16) for all k, n which are allowed by the requirement that (16) be overdetermined. Thus it is essential to be able to compute $[f^i]_0$ for as many i's as possible in reasonable time.

This method was used by Batyrev and van Straten [BvS] and has been polular ever since. More recently, Batyrev and Kreuzer used this approach in [BK] to determine the Picard-Fuchs operators for some new families of Calabi-Yau threefolds with Picard number 1. They succeeded in 28 cases out of 68. Using our algorithm were able to compute 4 more Calabi-Yau operators. Although we knew up to 300 constant terms, in many cases the recurrence did not show up. The operators and their corresponding polynomials are:

#24:

$$f_{24} = \tfrac{1}{T} + Y + \tfrac{T}{X} + \tfrac{ZT}{X} + \tfrac{ZT}{XY} + \tfrac{1}{Z} + \tfrac{X}{Z} + \tfrac{Y}{ZT} + \tfrac{X}{ZT} + \tfrac{XY}{ZT} + \tfrac{Y}{T} + \tfrac{T}{Y} + \tfrac{T}{XY} + \tfrac{Y}{X} + \tfrac{1}{X} + \tfrac{ZT}{Y} + T + \tfrac{1}{Y} + X + \tfrac{X}{T} + \tfrac{1}{ZT} + Z + \tfrac{Z}{Y}$$

$$\begin{aligned}
D_{24} = {}& 97^2\,\theta^4 + 97z\theta\left(-291 - 1300\,\theta - 2018\,\theta^2 + 1727\,\theta^3\right)\\
&+z^2\left(-2709792 - 10216234\,\theta - 16174393\,\theta^2 - 13428812\,\theta^3 - 1652135\,\theta^4\right)\\
&+z^3\left(-138000348 - 443115594\,\theta - 568639497\,\theta^2 - 364126194\,\theta^3 - 81753435\,\theta^4\right)\\
&+z^4\left(-3049275024 - 8869415520\,\theta - 10006378570\,\theta^2 - 5423394464\,\theta^3 - 1175502862\,\theta^4\right)\\
&+z^5\left(-38537290992 - 103964102350\,\theta - 106108023451\,\theta^2 - 50507429234\,\theta^3 - 9726250397\,\theta^4\right)\\
&+z^6\left(-308040167808 - 781527778884\,\theta - 733053660150\,\theta^2 - 312374434824\,\theta^3 - 52762935894\,\theta^4\right)\\
&+z^7\left(-1619360309088 - 3901093356168\,\theta - 3399527062044\,\theta^2 - 1313199235080\,\theta^3 - 195453433908\,\theta^4\right)\\
&-144z^8\,(\theta+1)\left(3432647479\,\theta^3 + 22487363787\,\theta^2 + 50808614711\,\theta + 38959393614\right)\\
&-432z^9\,(\theta+2)\,(\theta+1)\left(1903493629\,\theta^2 + 10262864555\,\theta + 14314039440\right)\\
&-438048z^{10}\,(1862987\,\theta + 5992902)\,(\theta+3)\,(\theta+2)\,(\theta+1)\\
&-368028363456z^{11}\,(\theta+1)\,(\theta+2)\,(\theta+3)\,(\theta+4)
\end{aligned}$$

#39:

$$\begin{aligned}
f_{39} = {}& \tfrac{X}{YZ} + \tfrac{ZT}{XY} + ZT + T + \tfrac{Y}{X} + \tfrac{Z}{X} + \tfrac{Y}{XT} + Y + \tfrac{XY}{ZT} + \tfrac{X}{Z} + \tfrac{Y}{T} + Z + \tfrac{1}{T} + X + \tfrac{X}{Y} + \tfrac{ZT}{Y} + \tfrac{X}{ZT}\\
&+\tfrac{Y}{ZT} + \tfrac{1}{X} + \tfrac{1}{Y} + \tfrac{ZT}{X} + \tfrac{T}{Y} + \tfrac{1}{Z}
\end{aligned}$$

$$\begin{aligned}
D_{39} = {}& 16\,\theta^4 - 4z\theta\,\left(12 + 53\,\theta + 82\,\theta^2 + 2\,\theta^3\right)\\
&+z^2(-5120 - 18308\,\theta - 26199\,\theta^2 - 18410\,\theta^3 - 4895\,\theta^4)\\
&+z^3(-143808 - 430092\,\theta - 497452\,\theta^2 - 272424\,\theta^3 - 60679\,\theta^4)\\
&+z^4(-1478544 - 3987101\,\theta - 4034628\,\theta^2 - 1870838\,\theta^3 - 344527\,\theta^4)\\
&-z^5\,(\theta+1)\left(1076509\,\theta^3 + 5847783\,\theta^2 + 11226106\,\theta + 7492832\right)\\
&-2z^6\,(\theta+2)\,(\theta+1)\left(944887\,\theta^2 + 4249317\,\theta + 5045304\right)\\
&-3328z^7\,(518\,\theta + 1381)\,(\theta+3)\,(\theta+2)\,(\theta+1)\\
&-621920z^8\,(\theta+1)\,(\theta+2)\,(\theta+3)\,(\theta+4)
\end{aligned}$$

#41:

$$\begin{aligned}
f_{41} = {}& XT + YZ + ZT + \tfrac{1}{ZT} + T + \tfrac{1}{XZ} + \tfrac{1}{YZ} + \tfrac{1}{XY} + \tfrac{1}{XT} + XZT + \tfrac{1}{YT} + YZT + \tfrac{1}{XYZ} + \tfrac{1}{XYT}\\
&+\tfrac{1}{XZT} + \tfrac{1}{YZT} + XYZT + \tfrac{1}{XYZT} + Y + X + Z + \tfrac{1}{T} + \tfrac{1}{Z} + \tfrac{1}{X} + \tfrac{1}{Y}
\end{aligned}$$

$$\begin{aligned}
D_{41} = {}& 91^2\,\theta^4 + 91z\theta\left(-273 - 1210\,\theta - 1874\,\theta^2 + 782\,\theta^3\right)\\
&+z^2\left(-2649920 - 9962953\,\theta - 15227939\,\theta^2 - 11622522\,\theta^3 - 2515785\,\theta^4\right)\\
&+z^3\left(-110445426 - 348819198\,\theta - 432607868\,\theta^2 - 258678126\,\theta^3 - 59827597\,\theta^4\right)\\
&+z^4\left(-1915723890 - 5439732380\,\theta - 5901995820\,\theta^2 - 2998881218\,\theta^3 - 612043042\,\theta^4\right)\\
&+z^5\left(-18479595006 - 48522700563\,\theta - 47503242813\,\theta^2 - 21226829058\,\theta^3 - 3762840342\,\theta^4\right)\\
&+z^6\left(-110147546634 - 271941545379\,\theta - 244753624741\,\theta^2 - 98210309094\,\theta^3 - 15265487382\,\theta^4\right)\\
&+z^7(-422269162452 - 991829482602\,\theta - 831965057114\,\theta^2 - 304487632282\,\theta^3 - 42103272002\,\theta^4)\\
&-2z^8\,(\theta+1)\left(39253400626\,\theta^3 + 275108963001\,\theta^2 + 654332416678\,\theta + 521254338620\right)\\
&-z^9\,(\theta+2)\,(\theta+1)\left(94987355417\,\theta^2 + 545340710193\,\theta + 799002779040\right)\\
&-1540z^{10}\,(43765159\,\theta + 149264765)\,(\theta+3)\,(\theta+2)\,(\theta+1)\\
&-2^2 35^2 7^2 11^2 11971z^{11}\,(\theta+1)\,(\theta+2)\,(\theta+3)\,(\theta+4)
\end{aligned}$$

#38:

$$\begin{aligned}
f_{38} = {}& \tfrac{X}{Y} + \tfrac{ZT}{Y} + Z + X + T + \tfrac{X}{Z} + Y + \tfrac{Z}{X} + \tfrac{YZ}{X} + \tfrac{1}{XZT} + \tfrac{Y}{XZT} + \tfrac{Y}{XT} + \tfrac{1}{ZT} + \tfrac{1}{Z} + \tfrac{1}{YZ} + \tfrac{T}{Y}\\
&+\tfrac{X}{YZ} + \tfrac{Y}{ZT} + \tfrac{1}{XT} + \tfrac{1}{X} + \tfrac{1}{Y} + \tfrac{Y}{X} + \tfrac{1}{T}
\end{aligned}$$

$$\begin{aligned}
D_{38} = {}& 102^2\theta^4 - 102z\left(204 + 911\,\theta + 1414\,\theta^2 + 116\,\theta^3\right)\\
&+z^2(-2663424 - 9947652\,\theta - 14508941\,\theta^2 - 9892670\,\theta^3 - 2596259\,\theta^4)\\
&+z^3(-67967496 - 206933112\,\theta - 239004708\,\theta^2 - 125234088\,\theta^3 - 25685301\,\theta^4)\\
&+z^4(-598491604 - 1608054100\,\theta - 1587508748\,\theta^2 - 687051032\,\theta^3 - 112357900\,\theta^4)\\
&+z^5(-2495389956 - 6085656898\,\theta - 5273754198\,\theta^2 - 1927713868\,\theta^3 - 254678692\,\theta^4)\\
&+z^6(-5385015134 - 11995897911\,\theta - 9101625228\,\theta^2 - 2758627602\,\theta^3 - 283337071\,\theta^4)\\
&+z^7(-5612134720 - 11209872916\,\theta - 7075746650\,\theta^2 - 1555791344\,\theta^3 - 86504770\,\theta^4)\\
&+12z^8\,(\theta+1)\left(7613560\,\theta^3 + 27844427\,\theta^2 - 51849552\,\theta - 134696600\right)\\
&+z^9\,(\theta+2)\,(\theta+1)\left(60585089\,\theta^2 + 495871401\,\theta + 595115780\right)\\
&-600z^{10}\,(10279\,\theta - 113205)\,(\theta+3)\,(\theta+2)\,(\theta+1)\\
&-6790000z^{11}\,(\theta+1)\,(\theta+2)\,(\theta+3)\,(\theta+4)
\end{aligned}$$

References

[D] E.J. Dyson, *Statistical Theory of The Energy Levels of Complex Systems*, Journal of Mathematical Physics, vol. 3, 1962.

[BK] Batyrev, Kreuzer, *Constructing new Calabi-Yau 3-folds and their mirrors via conifold transitions* `arXiv:0802.3376v2`

[BvS] V. Batyrev, D. van Straten *Generalized Hypergeometric Functions and Rational Curves on Calabi-Yau Complete intersections in Toric Varieties, 1993*

[K] D. Knuth *The Art of Computer Programming, volume 2: Seminumerical algorithms*, 3rd ed., Addison-Wesley, 1998.

[T] L.R. Turner *Inverse of the Vandermonde matrix with Appplications*, NASA Technical Note, 1969

[DvK] J. Duistermaat, W. van der Kallen, *Constant terms in powers of a Laurent polynomial*, Indagationes Math. **9**, (1998), 221-231.

[N] *NVIDIA CUDA C Programming Guide*

[SR] N. Szabo, R.Tanaka, *Residue Arithmetic and its application to computer technology, 1967*